UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Primož Tramte

Ogrodje za vizualizacijo prostorskih podatkov

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Primož Tramte

Ogrodje za vizualizacijo prostorskih podatkov

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Marko Robnik Šikonja

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Obstajajo številne javne baze podatkov z geografsko opredeljenimi statistikami. Njihovo poznavanje je koristno za strokovnjake, ki se s temi področji ukvarjajo, marsikdaj pa tudi za širšo javnost, saj geografsko opredeljeni podatki pričajo o uspešnosti določenih državnih in lokalnih politik in ukrepov ter pomagajo razložiti nekatere družbene pojave. Pregled takšnih podatkov in njihova dejanska javna dostopnost sta mogoča le z ustrezno spletno vizualizacijo.

Izdelajte odprtokodno ogrodje, ki bo omogočalo spletno vizualizacijo različnih geografsko opredeljenih statističnih podatkov. Pri zasnovi ogrodja potrebno pozornost posvetite enostavnosti rabe in fleksibilnosti pri prikazu različnih podatkov. Delovanje ogrodja praktično preizkusite na nekaj primerih zanimivih podatkov.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Primož Tramte, z vpisno številko **63050119**, sem avtor diplomskega dela z naslovom:

Ogrodje za vizualizacijo geografskih podatkov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Robnika Šikonje,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 4. avgusta 2014 Podpis avtorja:

Zahvaljujem se mentorju prof. dr. Marku Robniku Šikonji za vse koristne napotke in usmeritve pri izdelavi diplomske naloge. Zahvaljujem se svojim sodelavcem za nasvete pri programiranju. Zahvaljujem se tudi svojim staršem, bratu Matjažu Tramtetu ter Niki Plesec za vso moralno podporo in spodbudo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ASP	active server pages	aktivne spletne strani
CSS	cascading style sheets	kaskadne stilne podloge
GIS	geographic information system	geografski informacijski sistem
HTML	hyper text markup language	označevalni jezik za izdelavo spletnih strani
JSON	JavaScript simple object notation	JavaScript notacija enostavnega objekta
KML	keyhole markup language	jezil za označevanje besedila s pomočjo ključev
MVC	model view controller	model pogled kontroler
PDF	portable document format	standard za izmenjavo elektronskih prenosljivih dokumentov
SHP	shapefile	datoteka za izmenjavo prostorskih podatkov
XML	extensible markup language	razširljiv označevalni jezik

Povzetek

V sklopu diplomskega dela je bilo izdelano ogrodje za vizualizacijo geografskih podatkov. Namen naloge je bilo izdelati orodje, ki bo tudi povprečnim uporabnikom omogočalo nalaganje datotek in prikaz datotek z geografsko vezanimi podatki. Opisan je pomen vizualizacije podatkov in predstavljena so temeljna načela zajema in vizualizacije podatkov. Ogrodje je realizirano v obliki spletne aplikacije. Predstavljene so uporabljene tehnologije in arhitektura aplikacije ter interakcija med glavnimi komponentami aplikacije. Opisan je uporabniški vmesnik. Uporaba aplikacije je razložena na nekaj primerih. Za demonstracijo razvite rešitve smo uporabili geografske podatke o občinah v Sloveniji in statistične podatke o porazdelitvi dohodka in brezposelnosti.

Ključne besede: vizualizacija geografskih podatkov, ogrodje za vizualizacijo geografskih podatkov, občine v Sloveniji, spletna aplikacija

Abstract

We present a web application for visualizing geographical data. The purpose of it was to create a tool that would enable non-experts to load and visualize geographical data. We describe the principles of data visualization and present common approaches of capturing and visualizing data. Our presentation framework is developed as a web application. We describe technologies we used, architecture of the application, interactions between the application components, and the user interface. We demonstrate the use of the application through examples based on geographical data of Slovenian municipalities and statistical data of income distribution and unemployment.

Keywords: visualization of geographical data, framework for visualization of geographical data, Slovenian municipalities, web application

Poglavje 1 Uvod

V razvitem svetu razpolagamo z množico prosto dostopnih podatkov, ki jih je mogoče vizualizirati na različne načine, kar omogoča lažjo interpretacijo. Ljudje lažje interpretiramo grafični prikaz, kot pa podatke v tabeli. Cilj vizualizacije je torej posredovati informacije na razumljiv način in s tem omogočiti lažjo analizo podatkov.

Če pogledamo v zgodovino vizualizacije podatkov, se spomnimo miselnih vzorcev, diagramov in grafov, s katerimi se srečamo že zgodaj. V računalništvu so vizualizacije povezane z razvojem računalniške grafike. Pomembna je tudi vizualizacija geografskih podatkov. Na svetovnem spletu jih najdemo ogromno. Primer javne baze takih podatkov je spletna stran Supervizor za spremljanje izdatkov javnih institucij. Vizualizacije geografskih podatkov pogosto dajo vprašanja na odgovore, ki jih težko razberemo iz recimo tabelarične oblike. Karte je namreč lažje interpretirati in analizirati.

Na svetovnem spletu je mogoče najti veliko orodij za vizualizacijo geografskih podatkov ali tako imenovanih GIS orodij. Kratica GIS označuje geografske informacijske sisteme, ki so poleg vizualizacije namenjeni tudi zajemu, shranjevanju, vzdrževanju, obdelavi in analizi geografskih podatkov. Pri pregledovanju obstoječih tehnologij in rešitev za vizualizacijo sem naletel na izjemno zanimive primere, kot so prikaz zvočne onesnaženosti, vizualizacije migracij, toka denarja,... Ugotovil sem, da le malokatero orodje ponuja uporabniku nalaganje podatkov iz lastne datoteke ter nato prikaz naloženih podatkov. Nekatera orodja so kompleksna za neizobraženega uporabnika. Iz tega razloga smo se odločili, da razvijemo sistem, ki bo omogočal geografski prikaz podatkov, naloženih iz datoteke. V diplomski nalogi sem želel spoznati tehnologije, ki omogočajo vizualizacijo geografskih podatkov, z željo, da mi bodo koristile pri nadaljevanju študija in zaposlitve.

V drugem poglavju predstavimo pomen vizualizacije geografskih podatkov, zgodovino razvoja ter ključna načela zajema in predstavitve podatkov. Na kratko opišemo nekaj obstoječih rešitev. V tretjem poglavju opišemo tehnologije in orodja, ki smo jih uporabili pri razvoju. V četrtem poglavju predstavimo ogrodje za vizualizacijo geografskih podatkov, z opisom delovanja aplikacije in arhitekturo aplikacije. V petem poglavju opišemo praktične primere uporabe aplikacije ter njihovo analizo. V sklepnem poglavju predstavimo ugotovitve, povzamemo prednosti in slabosti rešitve ter predlagamo nadgradnje in razširitve.

Poglavje 2 Predstavitev geografskih podatkov

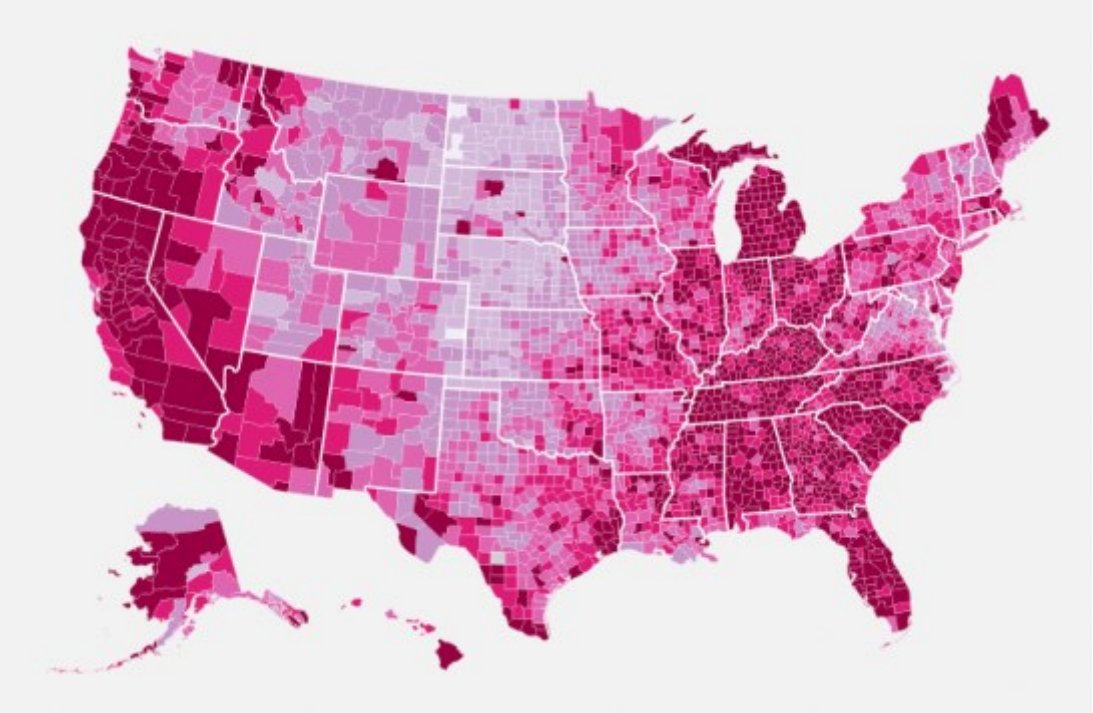
V tem poglavju opišemo pomen vizualizacije geografskih podatkov ter naštejemo nekaj primerov uporabe vizualizacije geografskih podatkov. Predstavimo temeljna načela zajema in predstavitve geografskih podatkov.

2.1 Pomen vizualizacije geografskih podatkov

Vizualizacija podatkov nam olajša razumevanje in analizo velikih količin podatkov. Namesto nepreglednih tabel predstavi podatke v lažje razumljivi obliki, da vidimo tudi sicer skrite informacije.

Leta 1854 je John Snow upodobil izbruh kolere v Londonu z uporabo točk glede na lokacije primerov. Njegova študija o porazdelitvi kolere je pripeljala do vira bolezni, to je do onesnažene vodne črpalke. Ko so črpalko zaprli, so omejili epidemijo. Medtem, ko so osnovni elementi topografije in kartografije obstajali že prej, je bil zemljevid Johna Snowa edinstven, saj je z uporabo kartografskih metod ne samo prikazal, temveč tudi analiziral geografsko odvisne pojave [2].

V novejši zgodovini lahko kot primer dobre uporabe vizualizacije podatkov navedemo prikaz nezaposlenosti po regijah v ZDA v letih od 2007 do 2009 [1]. Leta 2007 je bila nezaposlenost 4,4%. Do leta 2008 se je dvignila na 5,8%, ob koncu leta 2009 pa je bila že 9,8%. Katere regije imajo višjo brezposelnost kot druge? Katere regije so ostale nespremenjene? Na ta vprašanja je dala odgovor vizualizacija podatkov.



Slika 2.1: Vizualizacija brezposelnosti v ZDA leta 2009

Na sliki 2.1 so s temno barvo označene regije z visoko stopnjo nezaposlenosti, svetlejše regije pa imajo nižjo stopnjo brezposelnosti. Iz slike je razvidno, da so regije s stopnjo brezposelnosti, višjo od 10%, v veliki večini na zahodu in vzhodu države, medtem ko regije v osrednjem delu večinoma nimajo visoke brezposelnosti. Ti vzorci ne bi bili tako očitni, če bi namesto zemljepisne karte podatke imeli na voljo le v preglednici. Karta je tako omogočila lažje določanje območij za postavitev fundacij in ostalih kriznih centrov.

2.2 Prikaz geografskih podatkov

2.2.1 Zajemanje podatkov

Zajemanje podatkov [2] za vizualizacijo je dejansko pridobivanje dejstev ter njihova organizacija in fizično shranjevanje v podatkovne baze. V zadnjem desetletju so se razmišljanja in metode reševanja geografskih problemov zelo spremenila. Na voljo je veliko novih tehnologij:

- digitalna elektronika in omrežna tehnologija,
- tehnologija GPS in GIS,

- daljinsko zaznavanje in radarska tehnologija,
- uporaba radijskih povezav in mobilna telefonija,
- mobilna avtonomna elektronska merska oprema in robotizacija,
- prenosni in tablični računalniki,…

Za zajem podatkov potrebujemo strojno opremo za kartiranje (GPS, laserski merilnik razdalj) in računalnik za arhiviranje podatkov. Trend pri vizualizaciji geografskih podatkov je, da se kartiranje in analiza podatkov izvajata po sklopih. Zajem in vnašanje podatkov v sistem zahtevata veliko časa. Za to obstajajo različne metode:

- digitalizacija oziroma skeniranje obstoječih podatkov, natisnjenih na papir,
- neposreden vnos podatkov v sistem iz digitalnih sistemov za zbiranje podatkov,
- daljinsko zaznavanje iz senzorjev, povezanih z bazo,
- ortofoto posnetki,
- satelitsko daljinsko zaznavanje,…

Po vnosu podatkov v sistem je potrebno podatke urediti in odpraviti morebitne napake. Vektorski podatki morajo biti dovolj natančni, da jih lahko uporabimo za nadaljnje analize (na primer v cestnem omrežju morajo biti linije povezane z vozlišči na križiščih).

2.2.2 Predstavitev podatkov

Prikazani podatki [2] predstavljajo realne predmete (ceste, raba tal, nadmorska višina, drevesa, vodne poti,…) in njihove značilnosti. Realne predmete lahko razdelimo na enostavne (točkovni predmeti) in sestavljene (linijski elementi ali poligoni). V splošnem obstajata dve metodi, ki se uporabljata za shranjevanje podatkov v sisteme za geografsko vizualizacijo: rastrski in vektorski podatki. Vsi podatki so na koncu pretvorjeni v točke, linije in poligone. Po vsebini objekte delimo na:

- grafične: točka, linija, poligon,
- topološke: vozlišče, segmenti, robni poligon, površina,
- geografske: točkovni, linijski, ploskovni in objektni tip.

Na spletu se nahaja veliko število brezplačnih in nadvse uporabnih orodij za vizualizacijo podatkov:

- Atlas okolje [7]: orodje omogoča prikaz geografskih podatkov iz različnih kategorij (prostorske enote, merilna mesta, okolje, narava, tla, potresi,…) na karti Slovenije. Ima širok spekter funkcionalnosti, kot so npr. tiskanje karte, merjenje razdalj, točkovni presek, iskanje koordinat in parcel,…

- Geopedia [3]: v okviru portala Geopedia je mogoč prikaz podatkov kategorij, kot so kolesarstvo, pohodništvo, dogodki, turizem, vreme, študenti,… Na voljo so uporabne funkcionalnosti, kot so merjenje razdalje in površine, pretvornik koordinat, iskanje poti,…

- iObčina [5]: internetni GIS sistem, ki služi za iskanje, pregledovanje, analizo količin, merjenje razdalj in površin vseh vrst in tipov informacij v prostoru.

- iSlovenija [4]: GIS javnih in komercialnih prostorskih podatkov na območju Slovenije, namenjen poslovni uporabi.

- Prostorski informacijski sistem občin (PISO) [6]: omogoča enostaven vpogled v državne in občinske prostorske evidence, kot so: zemljiški kataster, prostorski plan, komunalni kataster, cestna infrastruktura, kataster stavb, ortofoto posnetki, poslovni subjekti in ostalo.

- KASPer [8]: uporabnikom omogoča prikazovanje demografskih statističnih podatkov po izbranih prostorskih enotah v kombinaciji z različnimi kartografskimi podlagami iz orodja Google Maps (Google zemljevidi).

Poglavje 3 Pregled uporabljenih tehnologij in orodij

Ogrodje za vizualizacijo geografskih podatkov je v obliki spletne aplikacije, ki temelji na strežniškem ogrodju ASP.NET MVC. To je ogrodje za razvoj spletnih aplikacij, ki deli aplikacijo na 3 plasti: modeli, pogledi in krmilniki. Za to ogrodje sem se odločil zaradi modularnosti in fleksibilnosti, ki nam jo ponuja. Podpira tehnologije, ki sem jih pri razvoju potreboval, ter omogoča dobro preglednost nad programsko kodo aplikacije, saj v ločene sklope deli podatkovne modele, pridobivanje podatkov in prikaz podatkov. Aplikacijo sem razvijal v razvojnem okolju Visual Studio. V tem poglavju so opisane uporabljene tehnologije.

3.1 .NET Framework

.NET Framework [9] je programsko ogrodje, razvito pri podjetju Microsoft. Omogoča hiter in enostaven razvoj tako obsežnih in zahtevnih kot tudi majhnih in enostavnih projektov. Vključuje obširno knjižnico in omogoča jezikovno interoperabilnost (vsak programski jezik lahko uporablja kodo, napisano v drugih jezikih) za nekaj različnih jezikov (C++, C#, Visual Basic, J#). Ogrodje ponuja širok spekter storitev, od povezovanja s podatkovnimi bazami, uporabe kriptografije, numeričnih algoritmov, omrežne komunikacije do razvoja uporabniških vmesnikov.

3.2 ASP.NET

ASP.NET (Active Server Pages) [10] je strežniško ogrodje za razvoj dinamičnih spletišč. Omogoča hiter razvoj dinamičnih spletnih strani. Razvili so ga pri podjetju Microsoft, z namenom omogočiti programerjem razvoj dinamičnih spletnih strani, spletnih aplikacij in spletnih storitev na podlagi platforme .NET. Ponuja:

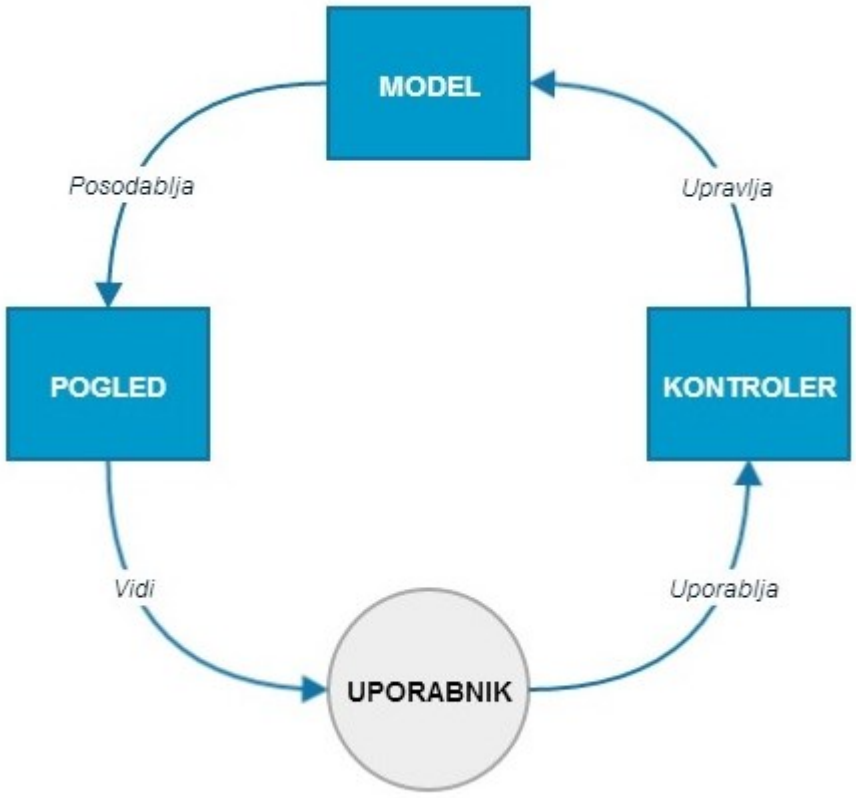
- osnovo za razvoj spletnih mest,
- upravljanje z uporabniškimi pravicami,

- množico dinamičnih spletnih gradnikov, ki jih lahko kombiniramo z uporabo (X)HTML, CSS in JavaScript-a v spletne strani in nadziramo na strežniški strani v .NET programski kodi (C# ali VisualBasic) in

- enostavno razširitev zgoraj opisanih komponent.

3.3 ASP.NET MVC

ASP.NET MVC [11] je platforma za razvoj spletnih aplikacij, ki izkorišča odlike ogrodja ASP.NET in implementira vzorec Model-View-Controller (model-pogled-krmilnik). Model predstavlja stanje določenega vidika aplikacije. Kontroler obdeluje interakcije z uporabnikom in ustrezno posodablja model, da ustreza spremembi stanja aplikacije, nato pa posreduje podatke pogledu. Pogled od kontrolerja prejme potrebne informacije in zgradi uporabniški vmesnik ter ga prikaže uporabniku. Interakcijo med komponentami MVC prikazuje Slika 3.1 3.1. Ločitev razvoja na te tri vidike omogoča večjo modularnost in fleksibilnost, hitrejši razvoj aplikacij ter njihovo lažje vzdrževanje.



Slika 3.1: Interakcija med komponentami MVC.

3.4 C#.NET

C# [12] je preprost, moderen, objektno usmerjeni programski jezik. Razvili so ga v podjetju Microsoft z namenom maksimalno izkoristiti zmogljivosti platforme .NET. Je nekakšna kombinacija programske moči jezika C++ in prednosti Jave in Visual Basic-a (objektna usmerjenost, ponovna uporaba kode, dogodki,…)

3.5 HTML

HTML (Hyper Text Markup Language) [13] je prevladujoč označevalni jezik za izdelavo spletnih strani. HTML elementi predstavljajo gradnike za vse spletne strani. HTML omogoča vgradnjo slik in drugih objektov (zvok, video) v dokument ter izdelavo interaktivnih obrazcev. Z uvedbo strukturne semantike za elemente (npr. naslovi, odstavki, seznami, tabele, povezave) ponuja strukturirane dokumente. Omogoča uporabo skriptnih jezikov (npr. JavaScript) za nadzor nad obnašanjem spletnih strani ter CSS (stilskih predlog) za določanje izgleda dokumentov. Razširitvi HTML, ki uporablja XML označevanje, pravimo XHTML.

HTML se je skozi čas spreminjal, širil in izpopolnjeval. Zadnja revizija HTML standarda, HTML5, prinaša izboljšave nadzora nad napakami v spletnih obrazcih, boljšo multimedijsko podporo (zvok, video), izboljšave strukturne semantike,...

3.6 XML

XML (Extensible Markup Language) [14] je označevalni jezik, ki določa množico pravil za kodiranje dokumentov v obliko, ki je razumljiva tako ljudem kot napravam. Razvit je bil z namenom, da bi stvari na spletu postale bolj enostavne, posplošene in uporabne. Je tekstovni podatkovni format z močno Unicode podporo za svetovne jezike. Čeprav je v osnovi namenjen za dokumente, ga pogosto uporabljamo za predstavitev poljubnih podatkovnih struktur, npr. za spletne storitve. Na podlagi sintakse XML so bili razviti številni dokumentni formati, npr. RSS, Atom, SOAP in XHTML.

3.7 CSS

CSS (Cascading Style Sheets) [15] so predloge, predstavljene v obliki preprostega slogovnega jezika, ki skrbi za predstavitev spletnih strani. Z njimi definiramo stil HTML/XHTML elementov s pravili prikaza strani. Določamo lahko barve, velikosti, odmike, poravnave, obrobe, položaj in vrsto drugih lastnosti, prav tako pa lahko nadziramo aktivnosti, ki jih

uporabnik izvaja nad elementi strani (npr. prekritje elementa z miško). CSS se najpogosteje uporablja na spletnih straneh, v jezikih HTML in XHTML, lahko pa se uporablja tudi v poljubnem XML dokumentu. Vpeljuje način ločevanja vsebine dokumenta (besedilo, slike, tabele, drugi elementi) od njegove oblike, s čimer:

- se zagotovi večja fleksibilnost in nadzor nad specifikacijo lastnosti,
- se zmanjša kompleksnost strani,
- lahko več strani uporablja iste oblikovne lastnosti, ne da bi bilo potrebno te lastnosti posebej pripisati vsaki strani,
- lahko posamezno stran prikažemo v različnih oblikah (npr. za prikaz na celotnem zaslonu in za prikaz v oknu).

CSS podpirajo vsi novejši spletni brskalniki.

3.8 JavaScript

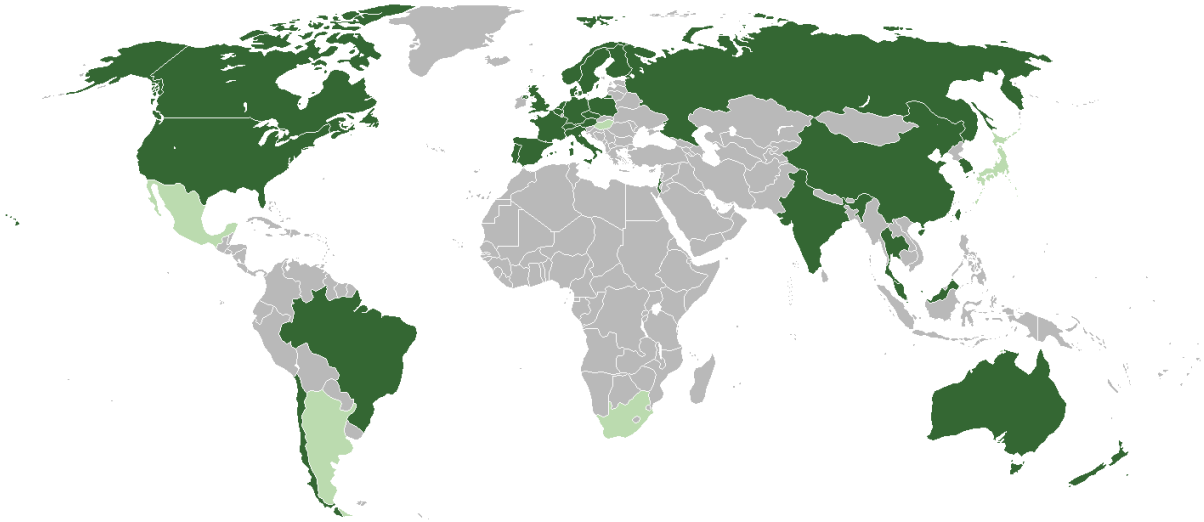
JavaScript [16] je objektni skriptni programski jezik, ki ga je razvilo podjetje Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani. Jezik je bil razvit neodvisno od jave, vendar si z njo deli številne lastnosti in strukture. JavaScript lahko sodeluje s HTML kodo in s tem poživi stran z dinamičnim izvajanjem. Ker gre za odprt jezik, ga lahko uporablja vsakdo, ne da bi za to potreboval licenco. JavaScript je podprt v vseh novejših spletnih brskalnikih.

3.9 jQuery

jQuery [17] je JavaScript knjižnica, namenjena poenostavitvi razvoja spletnih strani. Ideja te knjižnice je, da z manj kode naredimo več, kar predstavlja njen slogan “Write Less, Do More”. Prinaša enoten razvoj JavaScript kode za različne brskalnike, s čimer rešuje težave pri implementaciji iste funkcionalnosti za različne spletne brskalnike. Zaradi svoje razširljivosti ponuja odlično platformo za razvoj interaktivnih spletnih gradnikov. Podpirajo jo vsi novejši spletni brskalniki.

3.10 Google Maps JavaScript API

Google Maps JavaScript API [18] je JavaScript knjižnica, ki omogoča izrabo storitve Google Maps pri razvoju spletnih strani in aplikacij. Z nekaj vrsticami kode nam omogoča kreiranje interaktivne in natančne zemljepisne karte v naši aplikaciji, združljiva pa je tudi z ostalimi Googlovimi knjižnicami, kot so npr. Geocoding, Directions in StreetView. Slika 3.2 prikazuje primer uporabe knjižnice za vizualizacijo podatkov.



Slika 3.2: Primer uporabe Google Maps JavaScript API knjižnice za vizualizacijo podatkov.

3.11 Visual Studio

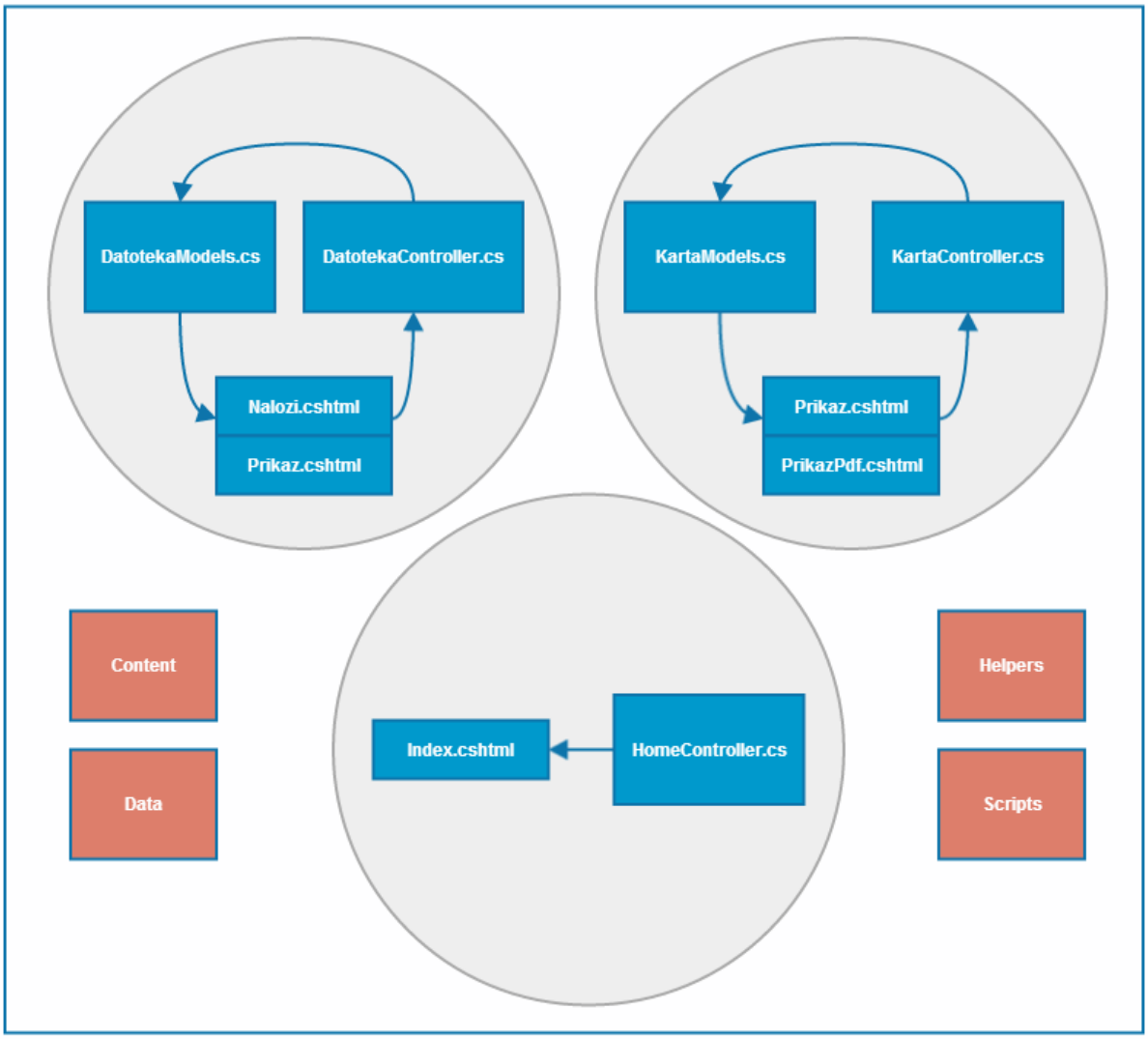
Visual Studio [19] je zbirka razvojnih orodij, ki ga je razvil Microsoft. Namenjeno je razvoju računalniških programov, spletnih strani in aplikacij ter spletnih servisov. Podpira platforme, kot so Windows API, Windows Forms, WPF, Windows Store, Microsoft Silverlight,… Ima vgrajeno podporo za kopico programski jezikov (C, C++, VB.NET, C#, F#,…), prav tako pa podpira tudi XML/XSLT, HTML/XHTML, JavaScript in CSS. Vključuje urejevalnik kode, ki podpira IntelliSense (samodejno dopolnjevanje kode in prikazovanje informacij o kodi), kakor tudi refaktoriranje kode (predelava strukture kode brez spreminjanja delovanja). Podpira vtičnike, ki izboljšujejo funkcionalnosti na vseh ravneh razvoja programske opreme.

Poglavje 4 Predstavitev ogrodja vizualizacij

V poglavju predstavimo arhitekturo aplikacije ter predstavimo njene glavne gradnike in značilnosti. Opišemo delovanje aplikacije in podrobnosti uporabniškega vmesnika.

4.1 Arhitektura aplikacije

Ogrodje za vizualizacijo smo sprogramirali v obliki spletne aplikacije, ki temelji na ogrodju ASP.NET MVC. Z uporabo programerskega vzorca MVC smo aplikacijo razdelili na 3 glavne komponente: modeli, kontrolerji in pogledi. Aplikacija vsebuje 3 kontrolerje, ki se nahajajo v direktoriju Controllers. Ti kontrolerji so DatotekaController.cs, HomeController.cs in pa KartaController.cs. Modeli se nahajajo v direktoriju Models v razredih DatotekaModels.cs in KartaModels.cs. Pogledi, ki jih uporabljajo kontrolerji, se nahajajo v direktoriju Views. Kontroler DatotekaController.cs uporablja pogleda Nalozi.cshtml in pa Prikaz.cshtml. Kontroler HomeController.cs uporablja pogled Index.cshtml, kontroler KartaController.cs pa uporablja pogleda Prikaz.cshtml in PrikazPdf.cshtml. Glavne komponente aplikacije in interakcije med njimi so prikazane na sliki 4.1. Poleg omenjenih direktorijev velja omeniti še Content (vsebuje datoteke, ki določajo izgled aplikacije), Data (vsebuje datoteko s koordinatami in naložene datoteke), Helpers (vsebuje razrede s pomožnimi metodami) in Scripts (vsebuje JavaScript datoteke za realizacijo funkcionalnosti, ki se izvajajo na strani odjemalca). Glavni deli programske kode naštetih komponent in interakcije med njimi so opisani v naslednjih poglavjih.



Slika 4.1: Glavne komponente aplikacije in interakcije med njimi.

4.1.1 Pridobivanje koordinat za določanje občin

Koordinate za določanje meja občin na karti Slovenije so brezplačno na voljo na spletnem portalu E-prostor in sicer v formatu shape (SHP). SHP [21] je geoprostorski vektorski format podatkov, ki ga uporabljajo geografski informacijski sistemi. Datoteka v formatu SHP prostorsko opisuje vektorske lastnosti: točke, črte in poligone, ki lahko predstavljajo jezera, reke, ceste,… Vsak element datoteke vsebuje atribute, ki ga opisujejo (npr. ime). Za potrebe spletne aplikacije je bilo podatke potrebno pretvoriti iz formata SHP v označevalni jezik (XML). To smo storili s pomočjo programa QGIS [22], ki je brezplačen odprtokodni program za geografsko vizualizacijo. Med drugim omogoča tudi shranjevanje datotek v velikem številu formatov, ki jih podpirajo različni geografski informacijski sistemi. S programom QGIS je mogoče pretvoriti datoteko iz formata SHP v označevalni jezik s ključi (KML) [23], ki temelji

na formatu XML. To smo storili tako, da smo v program naložili preneseno vektorsko datoteko SHP, ter pri nalaganju izbrali ustrezen koordinatni sistem. Nato smo podatke shranili v datoteko s formatom KML, ki smo ga določili ob shranjevanju. Datoteko v formatu KML smo odprli z brezplačnim urejevalnikom besedil Notepad++. Iz datoteke smo odstranili odvečne atribute za posamezne občine, saj za potrebe aplikacije potrebujemo le imena občin in zemljepisne koordinate njihovih meja. Poenostavljen primer predstavitve podatkov v formatu XML za posamezno občino je prikazan na sliki 4.2. Dokument smo po urejanju shranili v format XML.

```
<Placemark>
    <SimpleData name="OB_IME">VELIKE LAŠČE</SimpleData>
    <Polygon>
        <coordinates>
            14.540636279357813,45.875462850243487
            14.540639689545275,45.875048962860994
            14.540631478779284,45.874482065266733
            14.540620673444183,45.87423008199378
            14.54037801015472,45.873977166347167
            14.540355808621177,45.873545180139992
            14.540408815569586,45.873365436293668
            14.540501428594821,45.873068879596886
            14.539912770733633,45.875720877071814
            14.540636279357813,45.875462850243487
        </coordinates>
    </Polygon>
</Placemark>
```

Slika 4.2: Predstavitev podatkov občine v formatu XML.

Za branje koordinat iz datoteke formata XML smo v aplikaciji ustvarili naslednje razrede:

- razred Coordinates hrani podatke posameznih parov koordinat, ki določajo občino,
- razred Municipality hrani podatke posamezne občine (ime, vrednost podatka, seznam parov koordinat oziroma objektov tipa Coordinates),
- razred MunicipalitiesCoordinates hrani seznam občin oziroma objektov tipa Municipality.

V razredu HelperMethods smo razvili funkcijo GetCoordinatesFromXml, ki kot parameter sprejme ime datoteke in nato vrne objekt tipa MunicipalityCoordinates.

4.1.2 Branje naložene datoteke

Za branje podatkov iz naložene datoteke smo morali ustvariti model, ki bo hranil prebrane podatke. Za lažje kreiranje modela smo ustvarili dva pomožna razreda:

- razred RowR hrani seznam objektov tipa string, ki označujejo besedilo,
- razred SheetR hrani podatke za posamezni delovni zvezek, iz katerega smo brali podatke (ime, seznam vrstic tipa RowR).

S pomočjo teh razredov smo ustvarili razred DisplayFileModel z naslednjimi atributi:

- seznam naloženih datotek,
- ime trenutno uporabljane datoteke,
- seznam delovnih zvezkov trenutno uporabljanje datoteke (seznam objektov tipa SheetR).

V razredu HelperMethods smo razvili dve funkciji za branje podatkov iz datoteke, ReadXlsxFile in ReadXlsxFileByColumns. Obe funkciji vračata objekt tipa DisplayFileModel, ki vsebuje prebrane podatke iz datoteke, ki so prikazani na pogledu Prikaz.cshtml. Funkcija ReadXlsxFile kot parameter sprejme naziv datoteke in prebere celotno datoteko. Funkcija ReadXlsxFileByColumns kot parametre sprejme naziv datoteke, naziv delovnega zvezka, številko stolpca za občine in številko stolpca za podatke ter nato prebere podana stolpca delovnega zvezka.

4.1.3 Določanje podatkov za prikaz na karti

Določanje podatkov se izvaja na pogledu Prikaz.cshtml. Ko določimo datoteko za prikaz, se na kontrolerju DatotekaController.cs izvede akcija Prikaz, ki sprejeme naziv datoteke. Akcija pokliče prej omenjeno funkcijo ReadXlsxFile, ji poda naziv datoteke in napolni model DisplayFileModel s podatki. V akciji se izvede tudi pridobivanje že naloženih datotek na strežniku. Imena naloženih datotek se prav tako shranijo v model. Ko kontroler konča s pridobivanjem podatkov, poda model pogledu Prikaz.cshtml, ki se nahaja v mapi Datoteka. Pogled nato prikaže podane podatke. Za prikaz delovnih zvezkov datoteke smo uporabili spustne sezname. Funkcionalnosti ter obliko spustnih seznamov nam zagotavlja brezplačno ogrodje Bootstrap [24], ki skrbi tudi za obliko celotne aplikacije.

Določanje občin in podatkov v datoteki se izvede s pomočjo programskega jezika JavaScript in knjižnice imenovane jQuery, ki olajša manipulacijo komponent spletne strani. Vsakokrat, ko

uporabnik klikne v celico tabele, aplikacija preveri, ali smo v načinu za določanje občin ali podatkov ter ustrezno označi stolpec. Ko uporabnik klikne na gumb Zemljevid, se pokliče akcija Prikaz na kontrolerju KartaController.cs, ki ji podamo naslednje parametre (slika 4.3):

- naziv datoteke,
- ime delovnega zvezka,
- število stolpca, ki določa občine,
- število stolpca, ki določa podatke,
- število skupin,
- način razvrščanja v skupine in
- naziv podatka.

```
//click event for onMap button
$('#onMap').on('click', function () {
    //selected groups and mode
    _groups = parseInt($("#dataGroups option:selected").val());
    _mode = $("#dataMode option:selected").val();
    _dataName = $("#dataName").val();
    location.href = '@Url.Action("Prikaz", "Karta")' + '?naziv=' + '@Model.Filename' + '&imeZvezka=' +
       _sheetName + '&obcinaSt=' + _columnMunicipalities +                    '&podatekSt=' +
       _columnData + '&skupin=' + _groups + '&nacin=' + _mode + '&nazivPodatka=' + _dataName;
});
```

Slika 4.3: Klic akcije za prikaz podatkov na karti.

4.1.4 Prikaz podatkov na karti

Prikaz podatkov na karti Slovenije se izvede s klicem akcije Prikaz na kontrolerju KartaController. Ob klicu akcije se s pomočjo funkcije GetCoordinatesFromXml najprej izvede pridobivanje koordinat iz datoteke, ki je opisano v poglavju 4.1.1. Objekt, ki ga napolnimo s podatki, se imenuje coordsData. Ko v akciji pridobi koordinate občinskih meja, se s pomočjo funkcije ReadXlsxFileByColumns izvede branje podane datoteke s podatki, ki je opisano v poglavju 4.1.2. Objekt, ki ga napolnimo s podatki, se imenuje readData. Ko so podatki iz datoteke prebrani in je zgrajen objekt readData, se v zanki sprehodimo čez elemente tega objekta (predvidevamo, da objekt vsebuje naziv občine in podatke za posamezno občino). V zanki preverjamo, ali je prebran naziv občine vsebovan v objektu coordsData. Preverjanje se izvaja s pomočjo funkcije StringDistance, ki je definirana v razredu HelperMethods.cs. Funkcija kot parametra sprejme dve besedili in vrne podobnost med njima. S tem smo

zagotovili, da aplikacija upošteva tudi morebitne manjše napake v prebranih podatkih. Podobnost dveh besedil se računa po Levenshteinovi metodi [25]. Ob koncu izvajanja ene ponovitve zanke, se v objekt coordsData k ustrezni občini zapiše vrednost prebranega podatka. Ko se zanka izvede do konca, objekt coordsData vsebuje vrednosti prebranih podatkov za posamezne občine. Če podatka za določeno občino nismo uspeli prebrati, je za to občino vrednost podatka nastavljena na null. Prebrane podatke je sedaj potrebno dodati v model MunicipalitiesDataModel, ki ga bomo uporabili za prikaz podatkov na karti Slovenije.

Za kreiranje modela MunicipalitiesDataModel smo uporabili razrede:

- razred Coordinates hrani podatke posameznih parov koordinat, ki določajo občino,
- razred Municipality hrani podatke posamezne občine (ime, vrednost podatka, seznam parov koordinat oziroma objektov tipa Coordinates),
- razred MunicipalityGroup predstavlja eno skupino občin, torej hrani seznam objektov tipa Municipality ter vrednosti, ki določajo vrednostno območje skupine.

Atributi modela MunicipalitiesDataModel so:

- seznam skupin podatkov (seznam objektov tipa MunicipalityGroup),
- naziv datoteke,
- naziv delovnega zvezka,
- naziv podatka,
- najmanjši podatek,
- največji podatek,
- indeks stolpca za občine,
- indeks stolpca za podatke.

Izsek kode, ki predstavlja model MunicipalitiesDataModel, se nahaja na sliki 4.4.

```
//model for displaying data on map
public class MunicipalitiesDataModel {

    public MunicipalitiesDataModel()
    {
        this.MunicipalitiesGroups = new List<MunicipalityGroup>();
    }

    public List<MunicipalityGroup> MunicipalitiesGroups { get; set; }
    public string Filename { get; set; }
    public string Sheet { get; set; }
    public string DataName { get; set; }
    public decimal? Min {
        get { return this.MunicipalitiesGroups[0].Min; }
    }
    public decimal? Max {
        get { return this.MunicipalitiesGroups[this.MunicipalitiesGroups.Count - 1].Max; }
    }

    public int MunicipalitySt { get; set; }
    public int DataSt { get; set; }
}
```

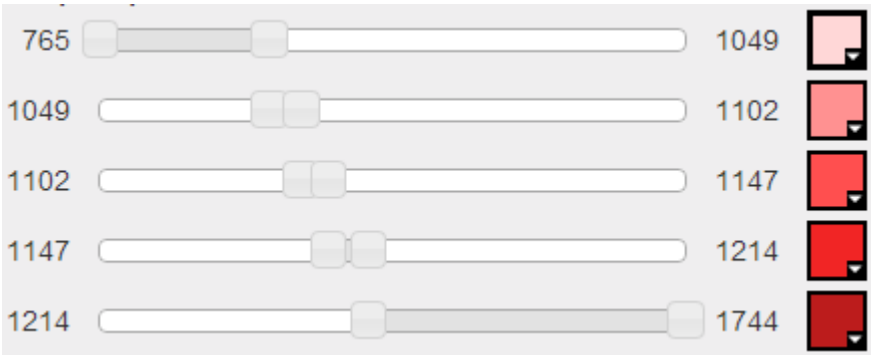
Slika 4.4: Model MunicipalitiesDataModel za prikaz podatkov na karti.

Način razvrščanja podatkov in število skupin so določeni v parametrih akcije. Načina za razvrščanje podatkov sta dva, po vrstnem redu in po kvantilih. Razvrščanje po vrstnem redu se izvede tako, da se najprej ustvari urejen seznam občin. Nato se izračuna velikost ene skupine podatkov glede na podano število skupin. Podatki se razporedijo v skupine. Za vsako skupino podatkov se izvede nova zanka, ki skupino napolni z ustreznimi podatki. Skupini podatkov določimo še največji in najmanjši podatek. Skupina se doda v seznam skupin modela. Zadnjo skupino podatkov napolnimo izven zanke, saj tako poskrbimo, da se skupina napolni z vsemi preostalimi podatki. Tudi ta skupina se doda v seznam skupin modela.

Razvrščanje podatkov po kvantilih razdeli podatke v skupine, kjer najmanjša in največja vrednost podatka enakomerno naraščajo. Najprej iz seznama podatkov pridobimo največji in najmanjši podatek, s pomočjo katerih izračunamo vrednost, za katero se bodo povečevala območja skupin. To vrednost hrani spremenljivka diff. V zanki za vsako skupino določimo vrednostno območje, torej največjo in najmanjšo vrednost podatka, ki je vsebovana v skupini. Ko določimo vrednostna območja skupin, v novi zanki pregledamo vse podatke objekta coordsData. Vsak podatek glede na vrednost razporedimo v ustrezno skupino podatkov in dodamo v seznam skupin modela.

Ob koncu izvajanja akcije model podamo pogledu Prikaz.cshtml, ki se nahaja v direktoriju Karta, nato pa se ta pogled prikaže v spletni aplikaciji. V pogledu iz modela preberemo podatke za naziv datoteke, naziv delovnega zvezka in naziv podatka. Glede na skupine, ki jih vsebuje

model, ustvarimo drsnike za spreminjanje območij posamezne skupine in transparentnosti. Poleg vsakega drsnika ustvarimo tudi kontrolo za izbiranje barve posamezne skupine. Drsnike in kontrolo za izbor barve ustvarimo s pomočjo jezika JavaScript in knjižnice jQuery (slika 4.5). V pogledu definiramo tudi območje, na katerem bo prikazana karta s podatki. Izsek kode HTML je prikazan na sliki 4.5.



Slika 4.5: Drsniki in kontrole za izbiranje barve

Karto prikažemo s pomočjo JavaScript knjižnice Google Maps JavaScript API. Model, ki smo ga podali pogledu, najprej za lažjo manipulacijo v jeziku JavaScript pretvorimo v objekt tipa JSON (JavaScript Simple Object Notation) [26]. Z objektom tipa JSON poenostavimo pregledovanje podatkov modela za prikaz podatkov na karti. V zanki pregledamo seznam občin, vsebovanih v modelu. Za vsako občino preberemo njen naziv, vrednost podatka ter koordinate, ki jo določajo. Iz prebranih koordinat zgradimo tako imenovani poligon (slika 4.6), ki predstavlja območje občine na karti. Iz naziva občine, vrednosti podatka in poligona ustvarimo nov objekt tipa JSON, ki ga shranimo v tabelo, imenovano municipalityPolygons. Podatke hranimo v tabeli zaradi možnosti spreminjanja vrednostnih območij in barv skupin (ta funkcionalnost je opisana v poglavju 4.2). Poligonu določimo proženje dogodka na klik, da se ob kliku nanj v oblačku prikažeta naziv občine in vrednost podatka. Poligonu na koncu določimo še mapo. Če na pogledu spremenimo območja, spremembe potrdimo s klikom gumba Zemljevid. Ob kliku v zanki se sprehodimo čez tabelo municipalityPolygons in posodobimo poligone. Pogled je mogoče shraniti v dokument PDF, kar je opisano v naslednjem poglavju.

4.1.5 Kreiranje dokumenta PDF

Kreiranje datoteke PDF se izvede ob kliku gumba Generiraj pdf. Pokliče se akcija PrikazPdf na kontrolerju KartaController.cs, ki ji podamo parametre:

- naziv datoteke,

- naziv delovnega zvezka,
- indeks stolpca občin,
- indeks stolpca vrednosti podatka za občine,
- število skupin,
- naziv podatka,
- transparentnost.

Podatke naziv datoteke, naziv delovnega zvezka, indeks stolpca občin, indeks stolpca vrednosti podatka za občine in naziv podatka nam zagotavlja model, ki ga uporablja pogled. Skupine podatkov, barve za skupine in transparentnost se pridobijo s pomočjo kode na sliki 4.6.

```
var groups = '';
var currentGroup = '';
var i = 0;
var min;
var max;
var color;
//loop čez legend info, da se napolni tabela
$('.legendInfo').each(function () {
    min = $('.min', $(this)).html();
    max = $('.max', $(this)).html();
    color = $('.color', $(this)).css('background-color');
    currentGroup = '&skupine[' + i + '].';
    groups += currentGroup + 'Max=' + max;
    groups += currentGroup + 'Min=' + min;
    groups += currentGroup + 'Color=' + color;
    i++;
});
```

Slika 4.6: Pridobivanje podatkov za kreiranje dokumenta PDF.

Akcija s pomočjo funkcij ReadXlsxFileByColumns in GetCoordinatesFromXml podobno kot akcija Prikaz napolni model MunicipalitiesDataModel ter nato iz pogleda PrikazPdf.cshtml ustvari datoteko PDF. To funkcionalnost nam zagotavlja paket Rotativa [29], ki ga namestimo z uporabo upravljalca paketov NuGet. Ob generiranju dokumenta PDF smo naleteli na težavo, saj dokument ni prikazoval karte. Težavo smo rešili z uporabo stikala window-status ok [28], ki smo ga podali ob klicu na koncu akcije. Stikalo window-status ok pomeni, da se dokument ne bo ustvaril, dokler pri izvajanju JavaScript kode na pogledu spremenljivka window.status ne bo imela nastavljene vrednosti na ok. Kodo smo torej priredili tako, da smo po nalaganju mape

počakali nekaj sekund in nastavili vrednost spremenljivke window.status na ok. Izsek kode se nahaja na sliki 4.7.

```
google.maps.event.addListenerOnce(map, 'idle', function () {
    setTimeout(
        function () {
            window.status = 'ok'
        }, 7000);
});
```

Slika 4.7: Izsek kode, ki reši težave s prikazom mape v dokumentu PDF.

4.2 Opis delovanja

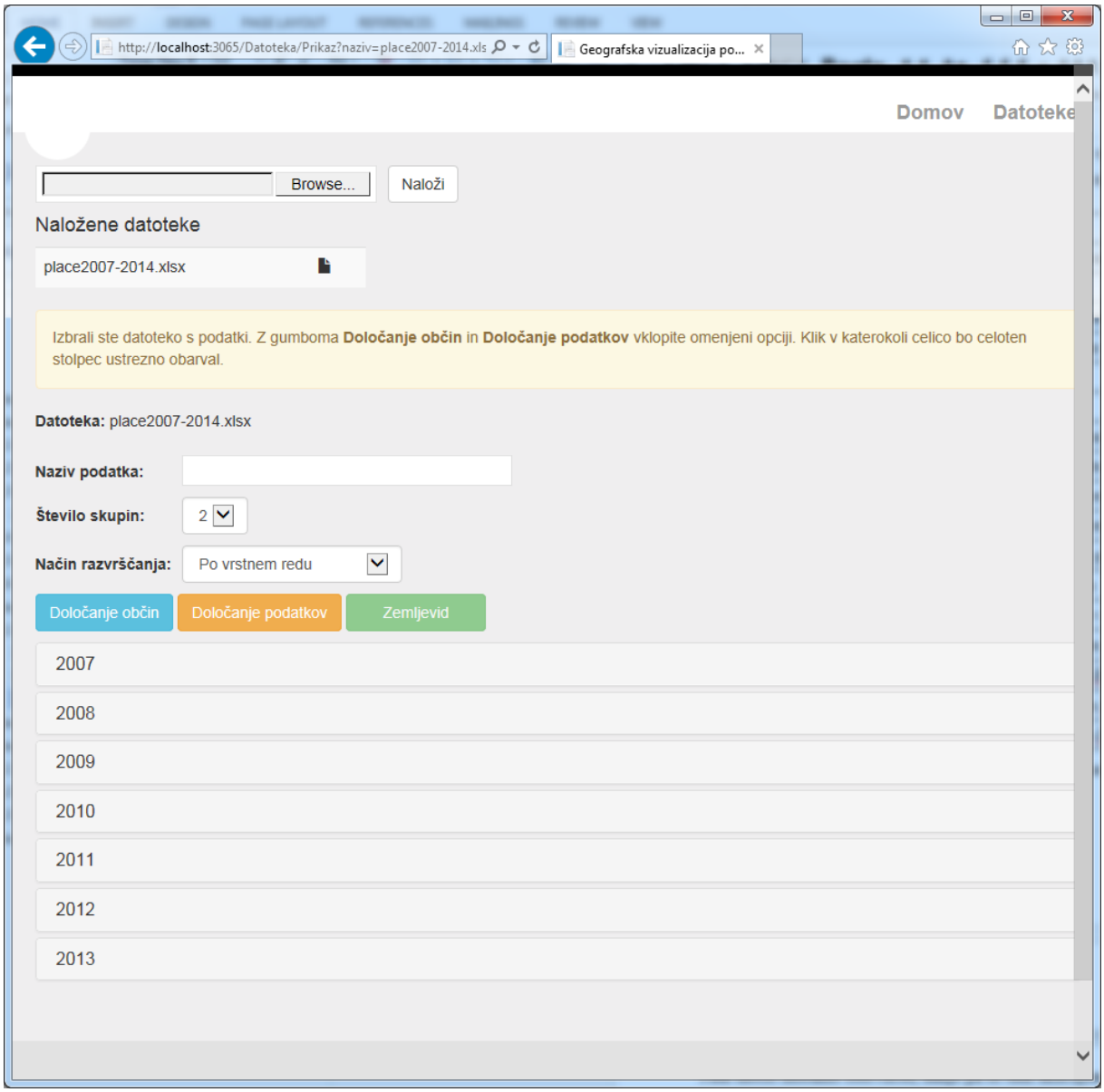
Navigacijska vrstica aplikacije vsebuje dve povezavi: Domov in Datoteke. Na prvi strani oziroma strani, na katero pridemo preko povezave Domov, se nahaja nekaj informacij o aplikaciji in o temi diplomskega dela, da uporabnika seznanimo s področjem, ki ga aplikacija pokriva.

Na strani Datoteke je realizirano nalaganje podatkov iz datoteke, ki jih je potem mogoče razvrstiti v skupine in prikazati na karti Slovenije. Na vrhu strani se nahaja kontrola za nalaganje datotek, pod njo pa se nahaja seznam že naloženih datotek. V manjšem barvnem okvirju so navodila, ki uporabnika vodijo skozi nalaganje datoteke in razvrščanje podatkov v skupine v primeru, da potrebuje pomoč.

Z gumbom Izberi datoteko se odpre okno za izbor datoteke, kjer izberemo poljubno datoteko s podatki (datoteka mora biti v formatu, ki ga podpira Microsoft Excel). Ko smo datoteko izbrali, svoj izbor potrdimo z gumbom Odpri. Sedaj je mogoče videti naziv datoteke znotraj kontrole za nalaganje datotek. S klikom gumba Naloži pošljemo datoteko na strežnik, kjer se iz datoteke preberejo podatki. Ko strežnik prebere podatke iz datoteke, se stran osveži ter ustrezno prilagodi (slika 4.8). Okvir za pomoč sedaj vsebuje navodila za nadaljevanje postopka vizualizacije. Pod okvirjem za pomoč se nahajajo nove kontrole:

- naziv pravkar naložene datoteke,
- kontrola za vnos naziva podatka,
- kontrola za izbor števila skupin podatkov,
- kontrola za izbor načina razvrščanja podatkov v skupine,
- gumbi za določanje občin, podatkov in prikaz zemljevida,

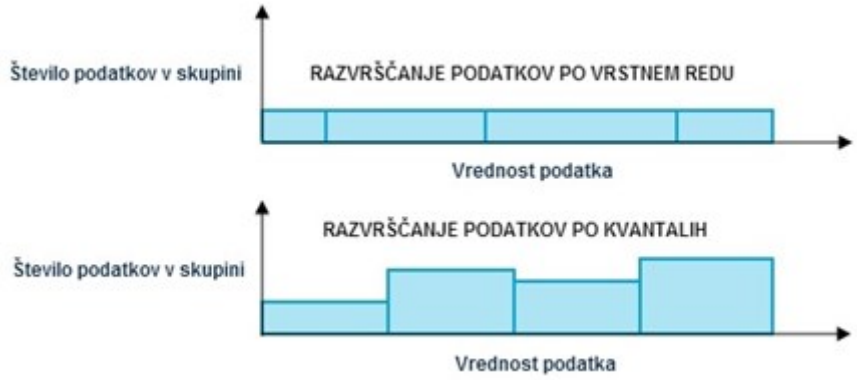
- spustni seznami vseh delovnih zvezkov, ki jih vsebuje naložena datoteka.



Slika 4.8: Zaslonski posnetek aplikacije po nalaganju datoteke.

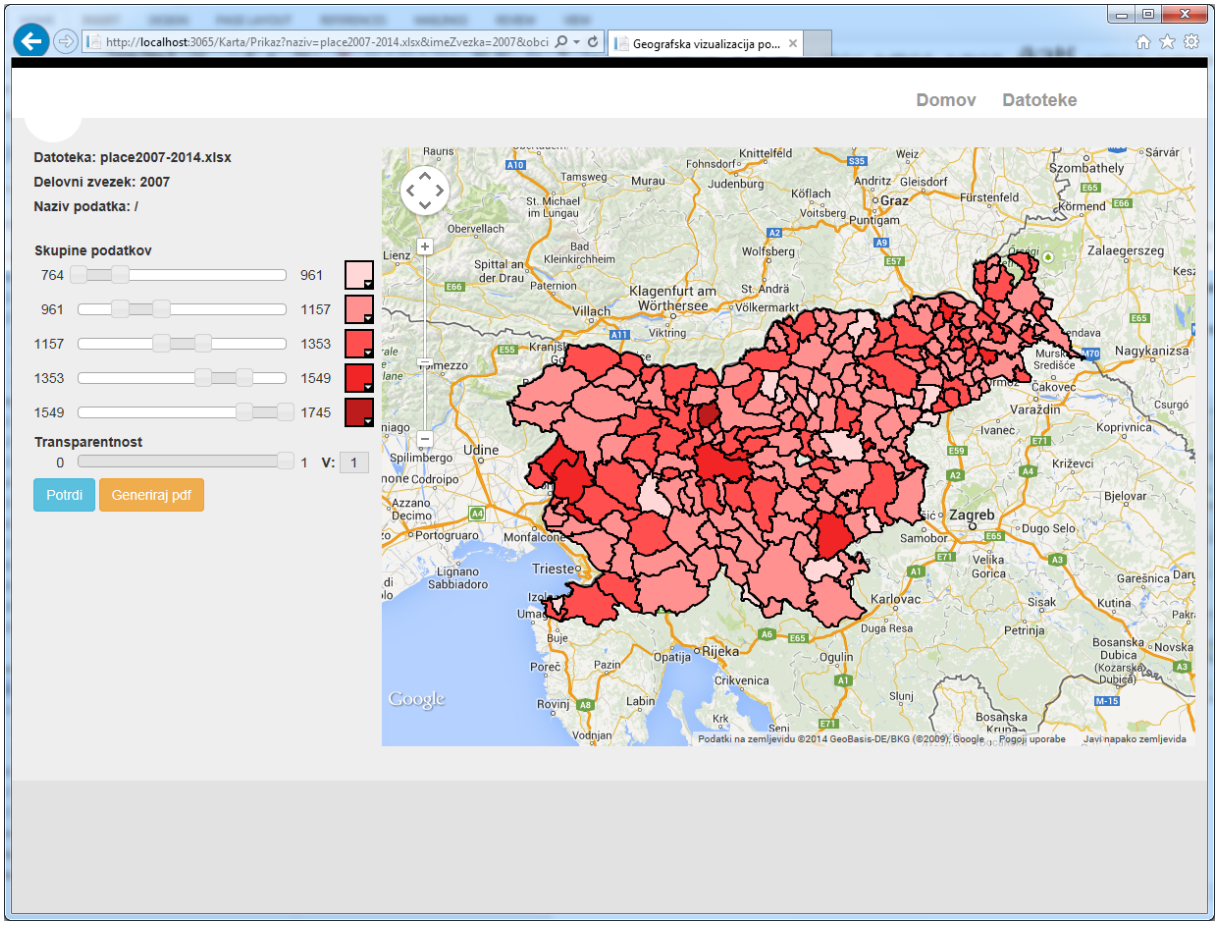
Spustni seznam posameznega delovnega zvezka odpremo s klikom na ime delovnega zvezka. Z gumbom za določanje občin vklopimo način za določanje občin v tabeli delovnega zvezka, ki smo ga pravkar odprli. Gumb se ob kliku ustrezno izboči, tako da uporabnik ve, da je aktiviran način za določanje občin. S klikom v celico, ki vsebuje ime občine, se z modro barvo označi celoten stolpec, saj aplikacija predvideva, da se v tem stolpcu nahajajo imena posameznih občin. Z gumbom za določanje podatkov vklopimo način za določanje podatkov. Tudi ta gumb se ob kliku ustrezno izboči, tako da uporabnik ve, da je aktiviran način za določanje podatkov. S klikom v celico, ki vsebuje podatek, se z oranžno označi celoten stolpec,

saj aplikacija predvideva, da se v tem stolpcu nahajajo podatki za posamezne občine. Ko smo označili občine in podatke, se omogoči gumb Zemljevid, posodobijo pa se tudi navodila v okvirju za pomoč. Sedaj je potrebno samo še vnesi naziv podatka ter določiti število skupin in pa način razvrščanja v primeru, da nam prednastavljene vrednosti ne ustrezajo (vnos naziva podatka je neobvezen). Število skupin nam pove, v koliko skupin se bodo razvrstili podatki glede na izbran način razvrščanja. Minimalno število skupin je 2, maksimalno pa 8. Načina razvrščanja podatkov sta 2, po vrstnem redu in po kvartalih. Razvrščanje po vrstnem redu razvrsti podatke v izbrano število skup po naraščajoči vrednosti podatka, zato imajo vse skupine podatkov enako število podatkov. Razvrščanje po kvantilih razvrsti podatke v izbrano število skupin, kjer se razmaki med najmanjšim in največjim podatkom v skupini enakomerno povečujejo. S tem se vrednostna območja skupin enakomerno razporedijo glede na vrednosti podatkov. Pri tem načinu razvrščanja je v določenih primerih, ko je razlika med vrednostma največjega in najmanjšega podatka velika, mogoče priti tudi do scenarija, da določena skupina nima podatkov. Slika 4.9 prikazuje oba načina razvrščanja podatkov v skupine.



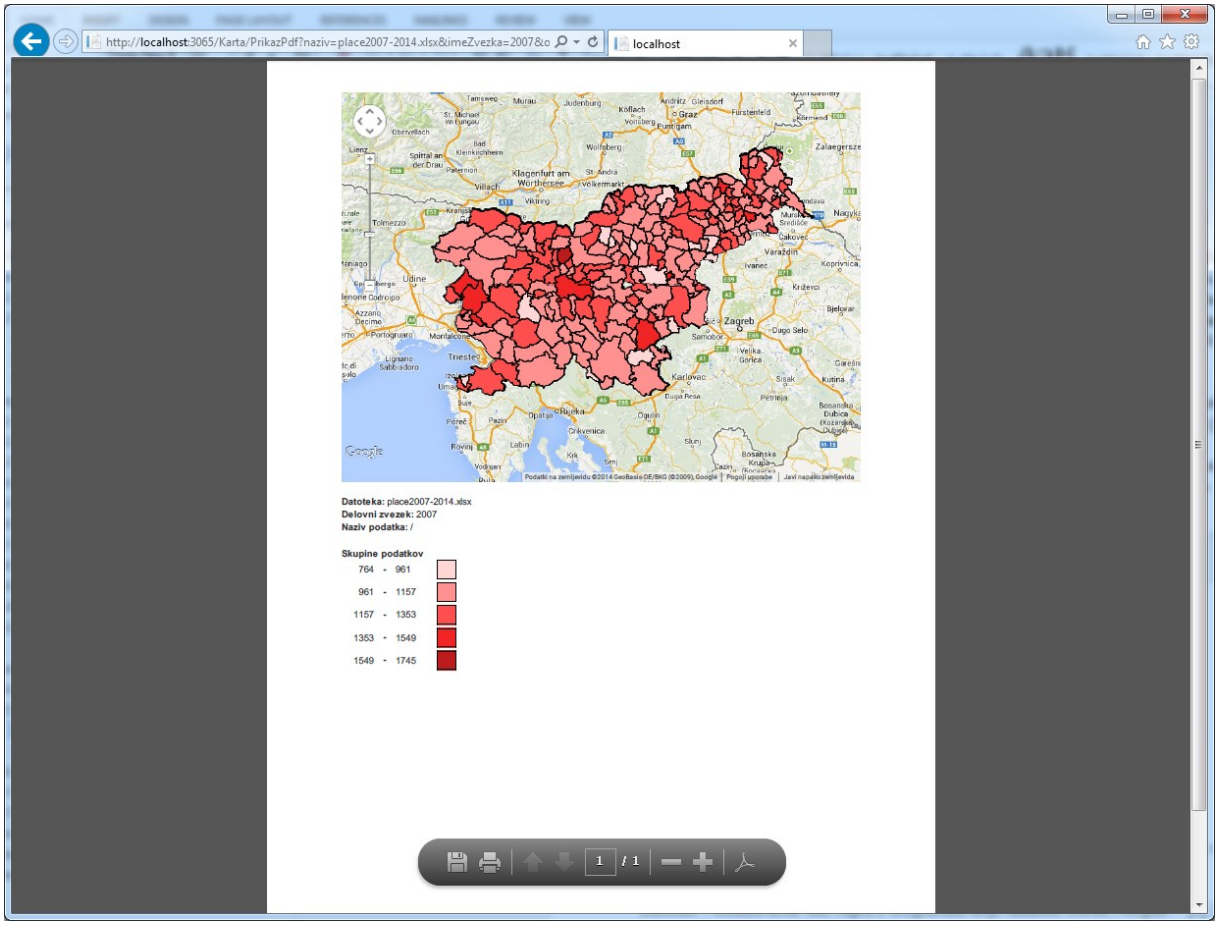
Slika 4.9: Prikaz obeh načinov razvrščanja podatkov v 4 skupine.

Ko smo določili občine in podatke, število skupin in način razvrščanja, je vse pripravljeno za prikaz podatkov na karti Slovenije. S klikom gumba Zemljevid nas aplikacija preusmeri na novo stran, kjer so izbrani podatki prikazani na karti (slika 4.10). Stran ima na levi strani izpisane informacije o datoteki, iz katere smo naložili podatke, pod temi informacijami pa so uporabniku na voljo kontrole za prilagajanje skupin podatkov, izbiro barve za posamezno skupino podatkov ter prilagajanje transparentnosti. Uporabniku je omogočena tudi možnost, da karto in prikazane podatke izvozi v datoteko PDF.



Slika 4.10: Prikaz podatkov na karti.

Kontrole za prilagajanje skupin so realizirane z uporabo drsnikov, s katerimi je mogoče spreminjati območje vrednosti posamezne skupine. Drsniki so prilagojeni tako, da ena skupina podatkov nikoli ne more vsebovati druge skupine podatkov. Ob vsakem drsniku je na desni strani kontrola za izbor barve, ki označuje to skupino. Ob kliku na to kontrole se prikaže kontrola za izbor barve. Pod drsniki za prilagajanje skupin podatkov se nahaja drsnik za določanje prosojnosti oziroma transparentnosti pobarvanih območij na karti. S spreminjanjem prosojnosti je na karti mogoče branje zemljepisnih podatkov na označenih območjih. Vse prilagoditve (območja vrednosti in barv skupin, transparentnost) se na karti odražajo po kliku gumba Potrdi, ki se nahaja pod kontrolo za določanje transparentnosti. Gumb Generiraj pdf ustvari in prikaže dokument PDF s podatki, kakor jih je uporabnik prilagodil na mapi. Izgled dokumenta je ustrezno prilagojen, tako da je na vrhu strani prikazana karta, pod njo pa se nahajajo informacije o datoteki, iz katere smo brali podatke, in informacije o skupinah podatkov (slika 4.11).

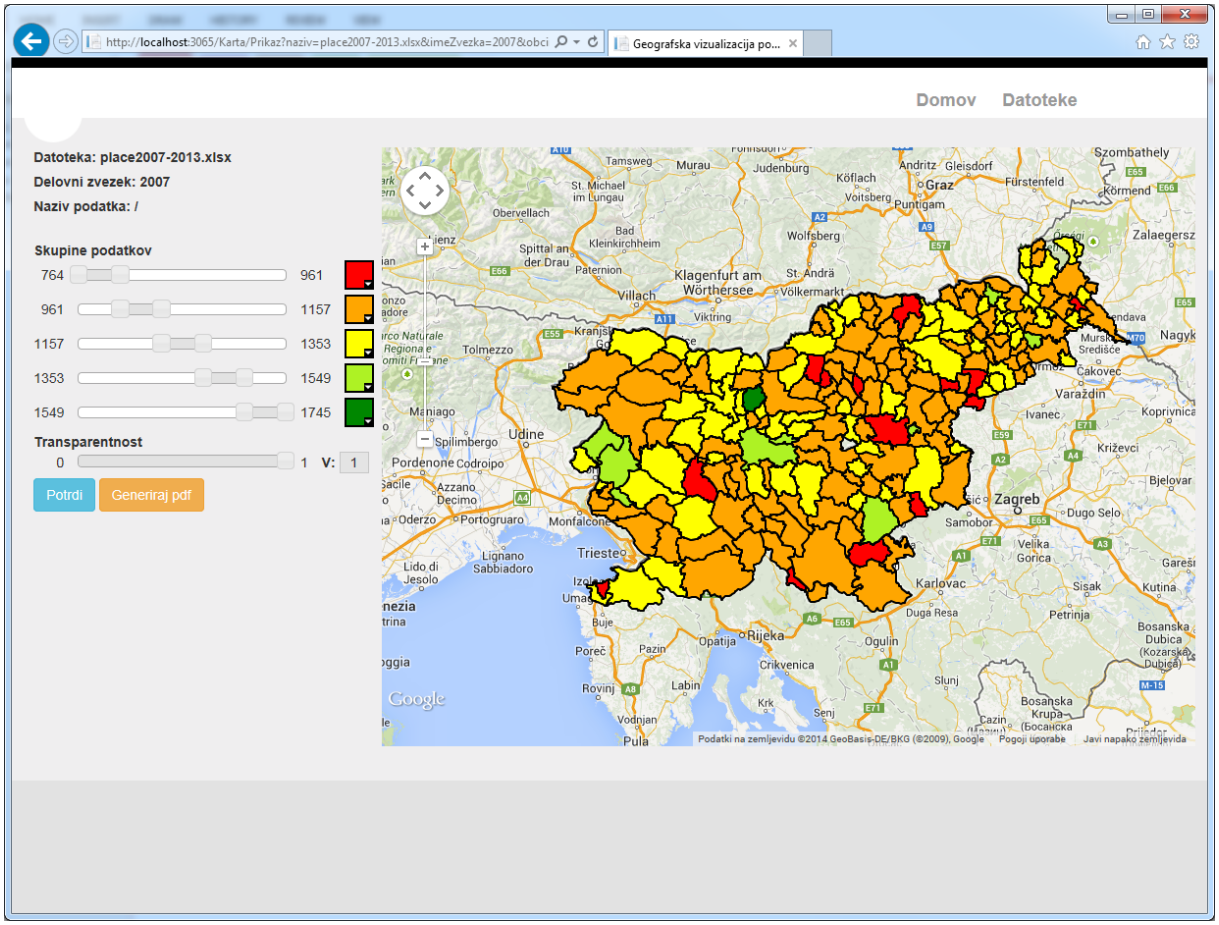


Slika 4.11: Prikaz karte v dokumentu PDF.

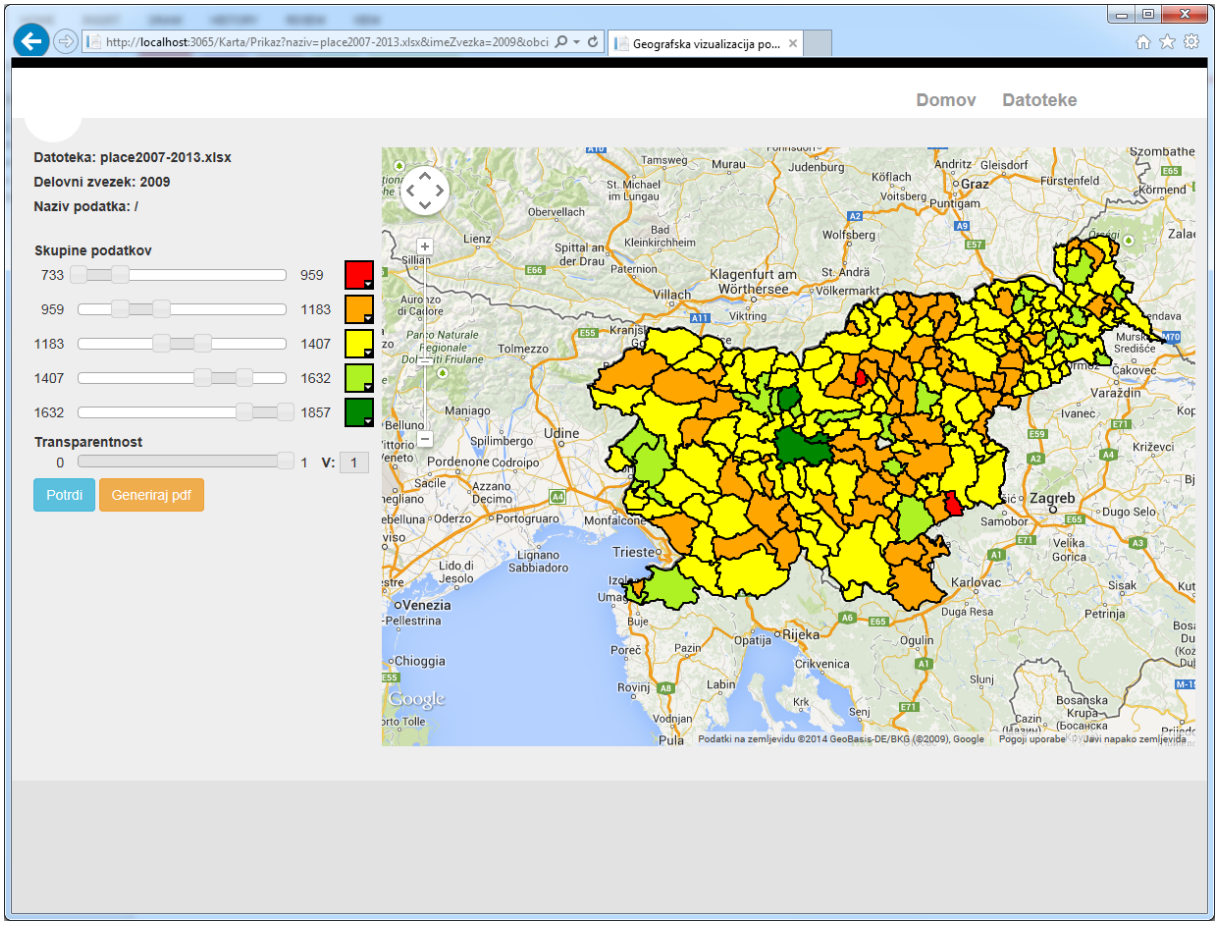
Poglavje 5 Primeri vizualizacij

V poglavju prikažemo nekaj primerov delovanja aplikacije. S pomočjo vizualizacije geografskih podatkov naložene podatke analiziramo ter predstavimo ugotovitve.

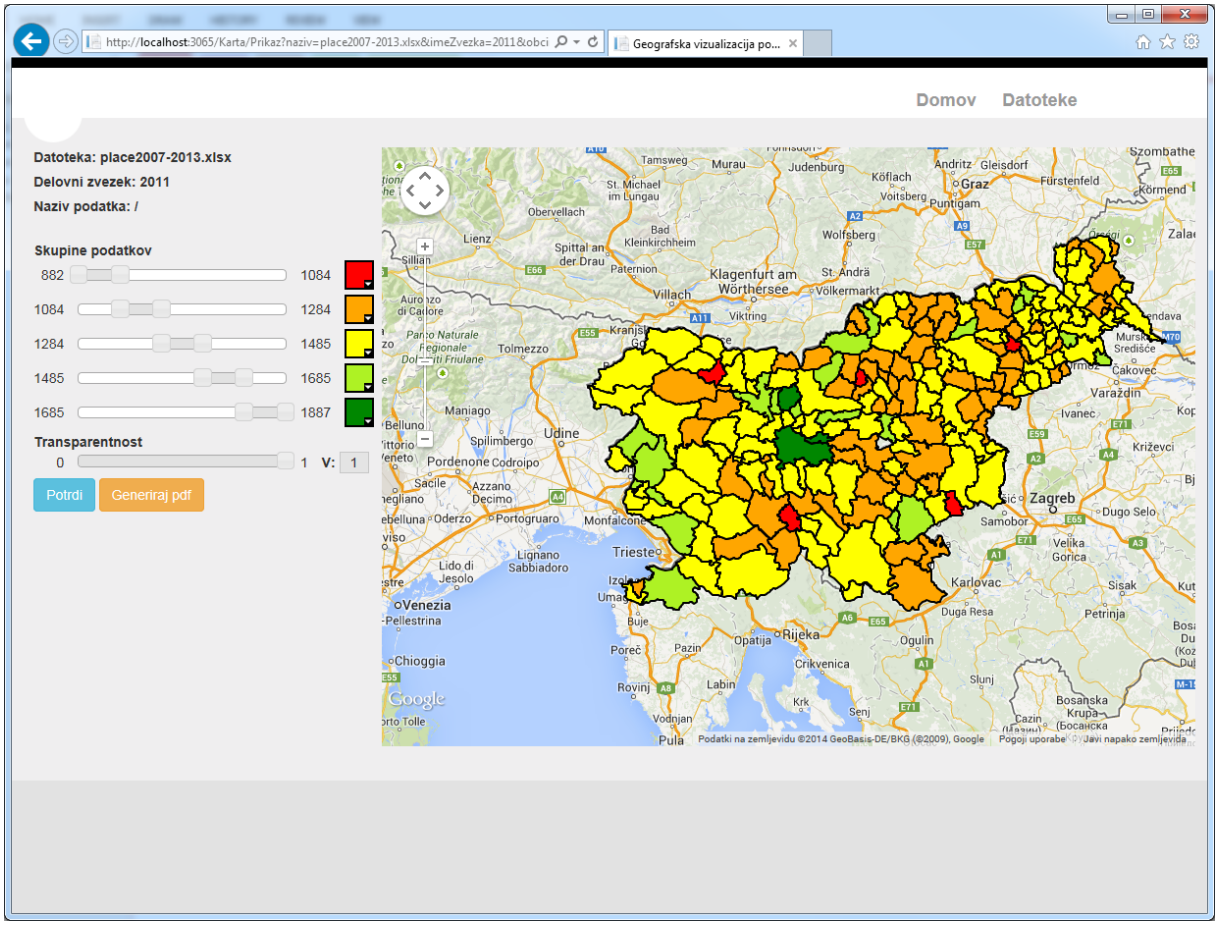
Za prvi primer delovanja spletne aplikacije smo izbrali prikaz podatkov za povprečne plače v Sloveniji v letih od 2007 do 2013. Podatke smo razvrstili v 5 skupin z načinom razvrščanja po kvartalih ter določili barve skupin, ki bodo jasno odražale razliko med občinami z najnižjimi in najvišjimi povprečnimi plačami. Predvidevali smo, da nam bo vizualizacija dala boljši vpogled v stanje povprečnih plač v slovenskih občinah pred in po recesiji ter tako odkrila, katere občine so bile bolj prizadete. Na spodnjih slikah si sledijo geografski prikazi povprečnih plač v slovenskih občinah v letih 2007 (slika 5.1), 2009 (slika 5.2), 2011 (slika 5.3) in 2013 (slika 5.4).



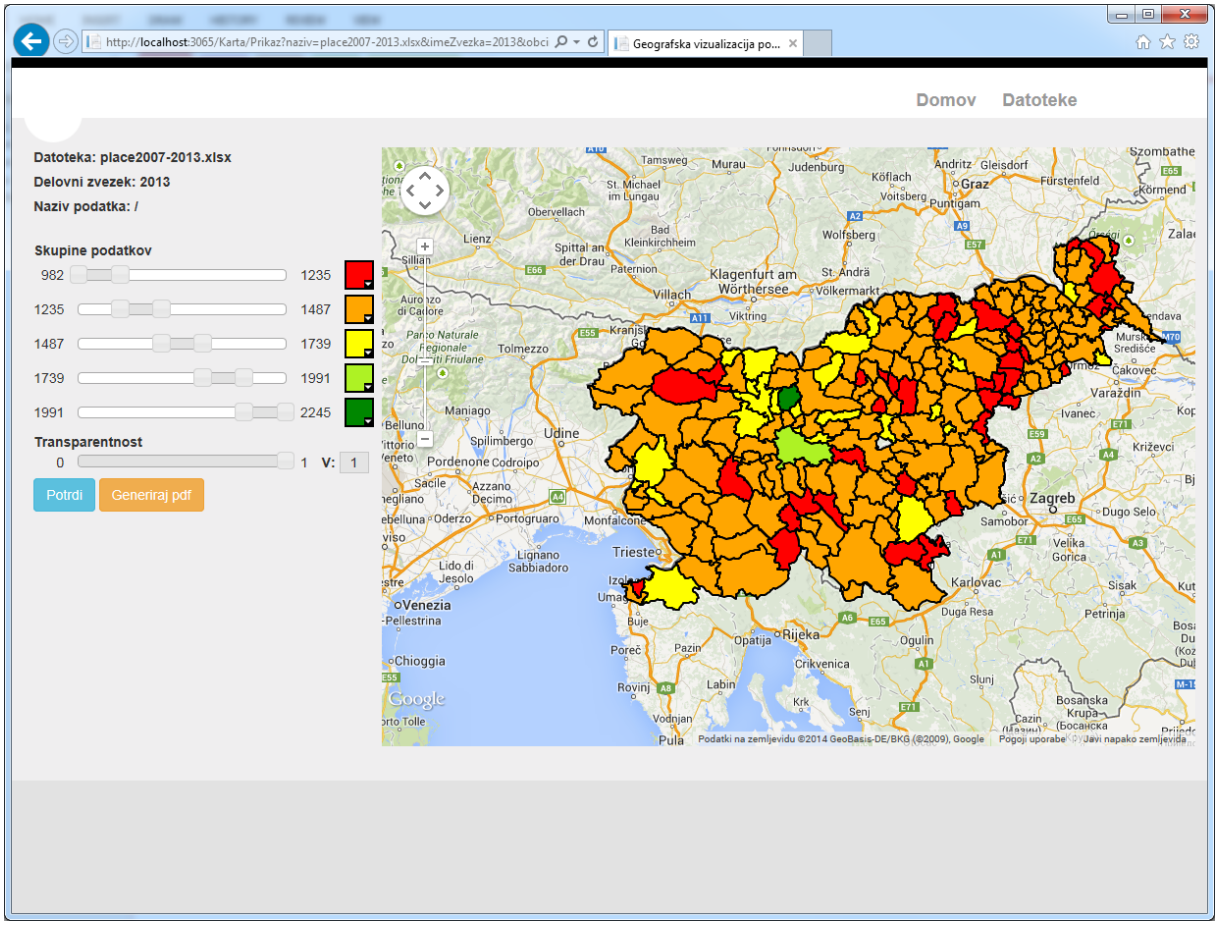
Slika 5.1: Povprečne plače za leto 2007.



Slika 5.2: Povprečne plače za leto 2009.



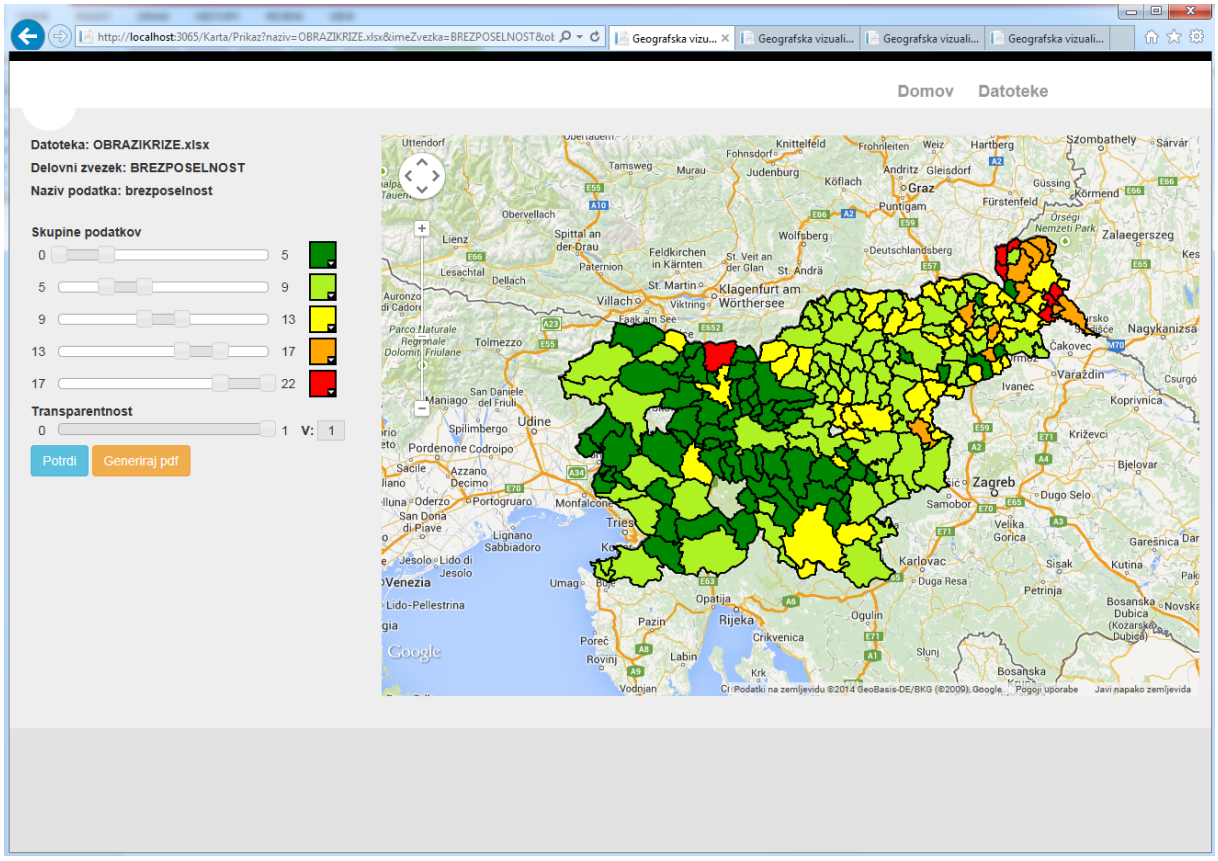
Slika 5.3: Povprečne plače za leto 2011.



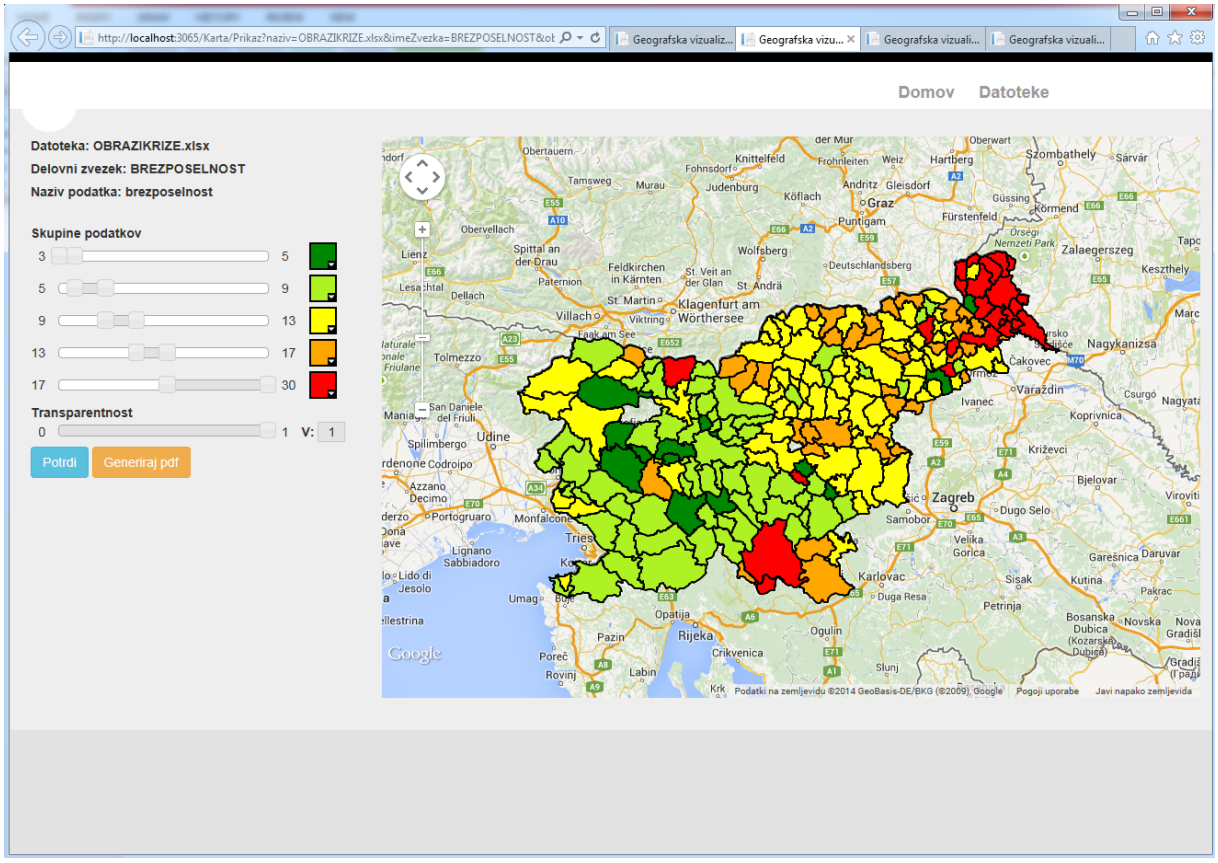
Slika 5.4: Povprečne plače za leto 2013.

Razvidno je, da število občin z nižjimi povprečnimi plačami narašča. Najbolj očiten je razviden v letu 2013. V letu vidimo tudi krizo, ki je prizadela mesto Maribor in okoliške občine. Občini Ljubljana in Cerklje na Gorenjskem sta skozi leta vedno ostali nad povprečjem, v letu 2013 pa je v prvi skupini podatkov ostala samo še občina Cerklje na Gorenjskem.

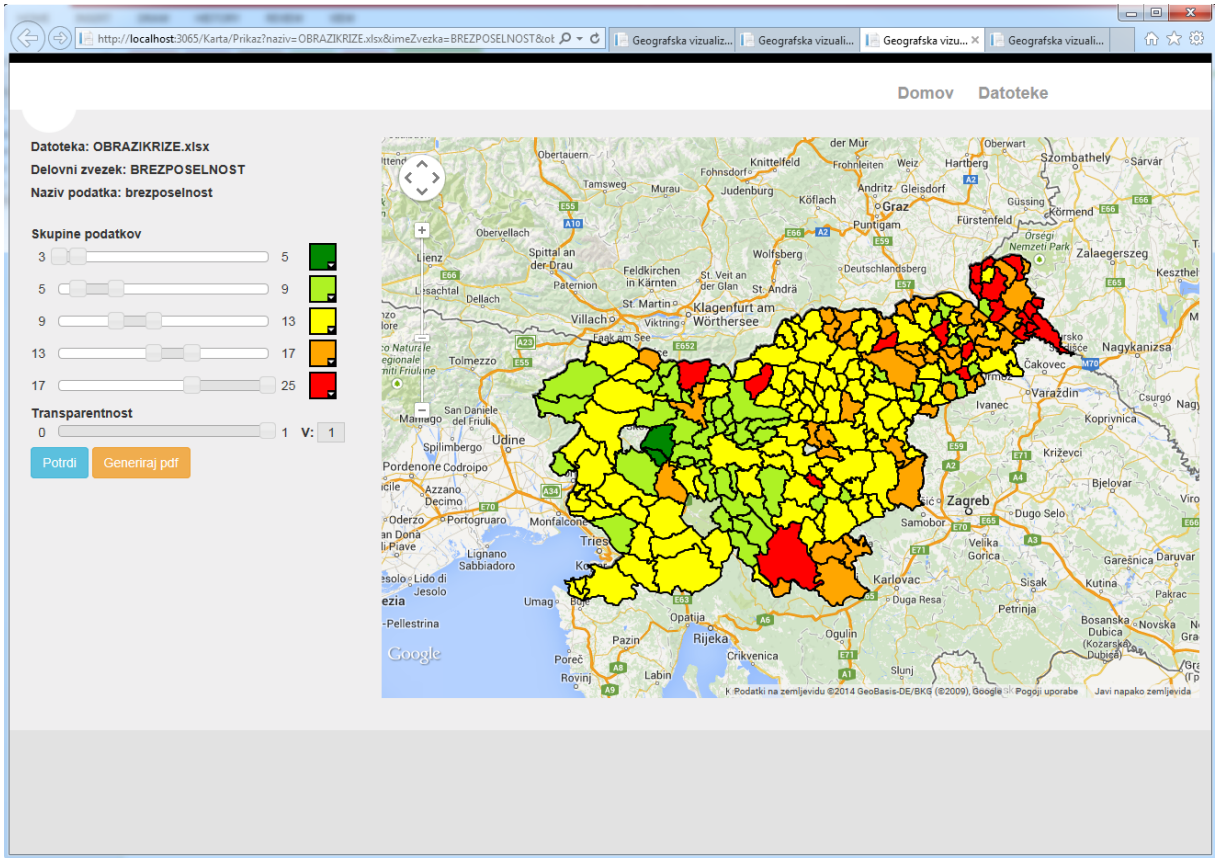
Za drugi primer smo izbrali prikaz podatkov o brezposelnosti v Sloveniji v letih od 2007 do 2013. Podatke smo razvrstili v 5 skupin z načinom razvrščanja po kvantilih ter določili barve skupin, ki bodo odražale razliko med občinami z najvišjo in najvišjo stopnjo brezposelnosti. Predvidevali smo, da bo iz prikazanih rezultatov razvidno, katere občine je recesija gospodarsko najbolj prizadela in katere občine so kljub recesiji uspele obdržati nivo brezposelnosti nad povprečjem. Na spodnjih slikah si sledijo geografski prikazi stopnje brezposelnosti v slovenskih občinah v letih 2007 (slika 5.5), 2009 (slika 5.6), 2011 (slika 5.7) in 2013 (slika 5.8).



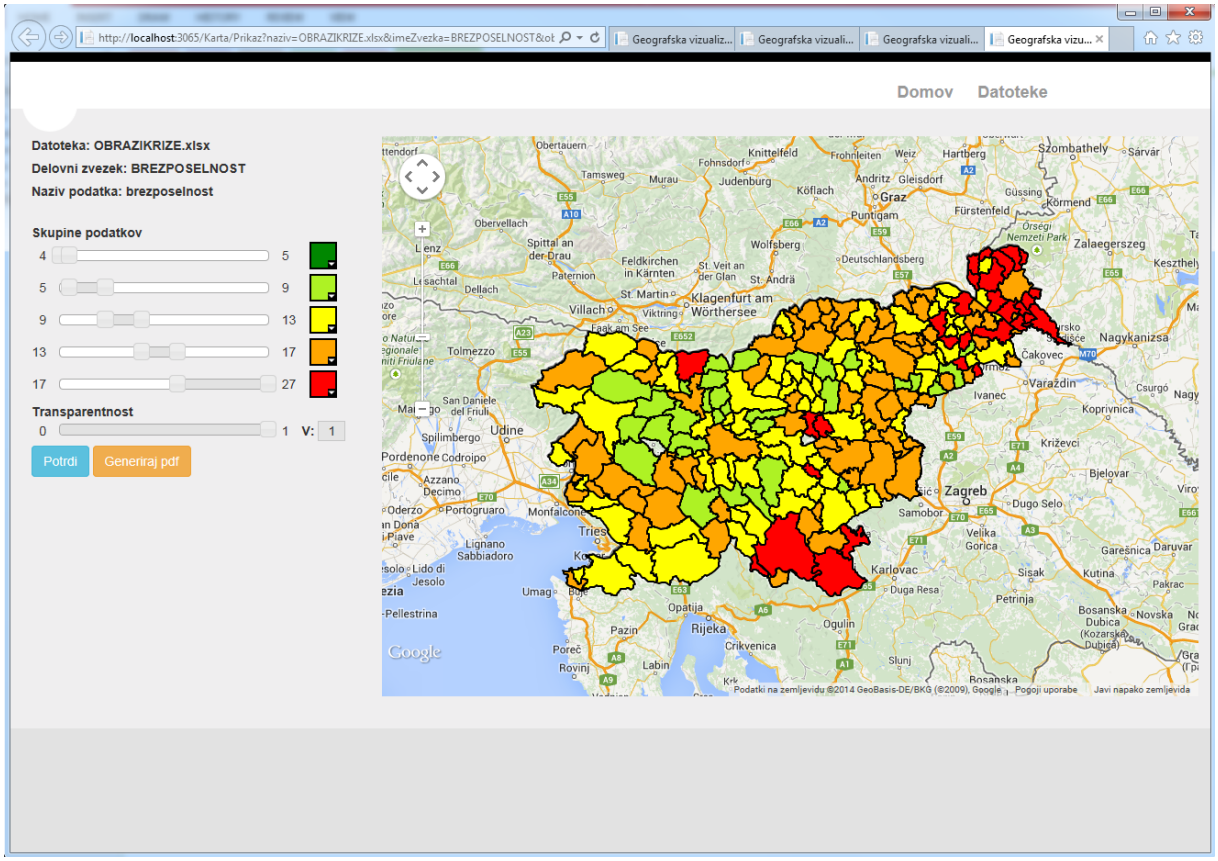
Slika 5.5: Stopnja brezposelnosti leta 2007.



Slika 5.6: Stopnja brezposelnosti leta 2009.



Slika 5.7: Stopnja brezposelnosti leta 2011.



Slika 5.8: Stopnja brezposelnosti leta 2013.

Iz podanih primerov vizualizacij je razvidno, da brezposelnost skozi leta vztrajno narašča v celotni državi. Povečanje števila brezposelnosti je bilo najvišje po letu 2007, ki sovpada z začetkom recesije. Najbolj prizadete so občine v severovzhodnem ter jugovzhodnem delu države, kjer se vidno odraža kriza zadnjih nekaj let. Leta 2013 v prvi skupini podatkov ni nobene občine več.

Poglavje 6 Sklepne ugotovitve

V diplomskem delu smo predstavili ključna načela vizualizacije geografskih podatkov in razvili rešitev, ki omogoča vizualizacijo prebranih podatkov. Opisali smo uporabljena orodja ter delovanje in arhitekturo aplikacije. Delovanje rešitve smo predstavili na praktičnih primerih, iz katerih je razvidna uporabnost rešitve.

Pri razvoju rešitve smo imeli na voljo mnogo programskih knjižnic za vizualizacijo podatkov, zato je bila izbira ustrezne precej težka. Zaradi enostavnosti in dobre dokumentacije sem se odločil za knjižnico Google Maps JavaScript API. Izbire ne obžalujem, saj se je skozi tok razvoja enostavnost uporabe te knjižnice znova in znova potrjevala.

Morebitna slaba stran rešitve je optimizacija hitrosti delovanja. Nalaganje koordinat in prikaz podatkov bi se pohitrila z uporabo statične mape, drugačnega načina shranjevanja koordinat,... Treba je upoštevati, da nekatere spremembe lahko potegnejo za sabo tudi izgubo funkcionalnosti. Dobra lastnost naše rešitve je njena uporabnost. Prikazani podatki so jasni in enostavni za analizo ter interpretacijo. Sistem bi lahko nadgradil s povezavo na katero od obstoječih baz geografskih podatkov, prikazovanjem toka podatkov in animacijami.

Menim, da smo izdelali kvalitetno in splošno uporabno rešitev. Zaradi enostavnosti uporabe je rešitev primerna za študente in vse, ki jih zanima vizualizacija geografskih podatkov.

Literatura

[1] Nathan Yau, *Visualize This*, Indianapolis: Wiley Publishing Inc., 2011.

[2] Geografski informacijski sistem. http://sl.wikipedia.org/wiki/Geografski_informacijski_sistem (dostop 29.6.2014)

[3] Geopedia. http://www.geopedia.si/#T184_x499072_y112072_s9_b4 (dostop 29.6.2014)

[4] iSlovenija. http://www.islovenija.si/gisapp/? (dostop 29.6.2014)

[5] iObčina. http://info.iobcina.si/iobcina3/index.php/sl (dostop 29.6.2014)

[6] Prostorski informacijski sistem občin. http://www.geoprostor.net/PisoPortal/Default.aspx? (dostop 29.6.2014)

[7] Atlas okolja. http://gis.arso.gov.si/atlasokolja/profile.aspx?id=Atlas_Okolja_AXL@Arso (dostop 29.6.2014)

[8] KASPeR. http://www.gis.si/kasper/si/index.html (dostop 29.6.2014)

[9] .NET Framework. http://en.wikipedia.org/wiki/.NET_Framework (dostop 29.6.2014)

[10] ASP.NET. http://en.wikipedia.org/wiki/Active_Server_Pages (dostop 29.6.2014)

[11] ASP.NET MVC. http://www.asp.net/mvc (dostop 29.6.2014)

[12] C Sharp (programming language). http://en.wikipedia.org/wiki/C_Sharp_(programming_language) (dostop 29.6.2014)

[13] HTML. http://sl.wikipedia.org/wiki/HTML (dostop 29.6.2014)

[14] XML. http://sl.wikipedia.org/wiki/XML (dostop 29.6.2014)

[15] CSS. http://sl.wikipedia.org/wiki/CSS (dostop 29.6.2014)

[16] JavaScript. http://sl.wikipedia.org/wiki/JavaScript (dostop 29.6.2014)

[17] jQuery. http://jquery.com/ (dostop 29.6.2014)

[18] Google Maps JavaScript API v3. https://developers.google.com/maps/documentation/javascript/examples/ (dostop 29.6.2014).

[19] Microsoft Visual Studio. http://en.wikipedia.org/wiki/Microsoft_Visual_Studio (dostop 29.6.2014)

[20] E-prostor. http://www.e-prostor.gov.si/ (dostop 29.6.2014)

[21] Shapefile. http://en.wikipedia.org/wiki/Shapefile (dostop 29.6.2014)

[22] QGIS. http://www.qgis.org/en/site/ (dostop 29.6.2014)

[23] Keyhole Markup Language. https://developers.google.com/kml/ (dostop 29.6.2014)

[24] Bootstrap. http://getbootstrap.com/ (dostop 29.6.2014)

[25] Levenshtein distance. http://en.wikipedia.org/wiki/Levenshtein_distance (dostop 29.6.2014)

[26] JSON. http://en.wikipedia.org/wiki/JSON (dostop 20.6.2014)

[27] C# Levenshtein. http://www.dotnetperls.com/levenshtein (dostop 29.6.2014)

[28] Wkhtmltopdf – Manual. http://madalgo.au.dk/~jakobt/wkhtmltoxdoc/wkhtmltopdf_0.10.0_rc2-doc.html (dostop 29.6.2014)

[29] Rotativa. https://github.com/webgio/Rotativa (dostop 29.6.2014)